IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of	:	
Michiko MATSUMOTO et al.	:	
Serial No. NEW	:	**Attn: APPLICATION BRANCH**
Filed November 14, 2003	:	Attorney Docket No. 2003-1649A

APPARATUS, METHOD AND PROGRAM
FOR CONTENTION ARBITRATION

CLAIM OF PRIORITY UNDER 35 USC 119

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

Applicants in the above-entitled application hereby claim the date of priority under the International Convention of Japanese Patent Application No. 2002-331905, filed November 15, 2002, as acknowledged in the Declaration of this application.

A certified copy of said Japanese Patent Application is submitted herewith.

Respectfully submitted,

Michiko MATSUMOTO et al.

By _____
Nils E. Pedersen
Registration No. 33,145
Attorney for Applicants

NEP/krg
Washington, D.C. 20006-1021
Telephone (202) 721-8200
Facsimile (202) 721-8250
November 14, 2003

日本国特許庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: ２００２年１１月１５日

出願番号
Application Number: 特願２００２－３３１９０５

[ST. 10/C]: ［ＪＰ２００２－３３１９０５］

出願人
Applicant(s): 松下電器産業株式会社

２００３年　９月　１日

特許庁長官
Commissioner,
Japan Patent Office 今井康夫

出証番号　出証特２００３－３０７１０００

【書類名】　特許願
【整理番号】　2037340046
【提出日】　平成14年11月15日
【あて先】　特許庁長官殿
【国際特許分類】　G06F 12/00　572
G06F 15/177　682
G06F 15/177　678
G06F 12/00　571
【発明者】
【住所又は居所】　大阪府門真市大字門真１００６番地　松下電器産業株式会社内
【氏名】　松本　美智子
【発明者】
【住所又は居所】　大阪府門真市大字門真１００６番地　株式会社松下ソフトリサーチ内
【氏名】　鈴木　良章
【特許出願人】
【識別番号】　000005821
【氏名又は名称】　松下電器産業株式会社
【代理人】
【識別番号】　100097445
【弁理士】
【氏名又は名称】　岩橋　文雄
【選任した代理人】
【識別番号】　100103355
【弁理士】
【氏名又は名称】　坂口　智康

【選任した代理人】

【識別番号】　100109667

【弁理士】

【氏名又は名称】　内藤　浩樹

【手数料の表示】

【予納台帳番号】　011305

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　9809938

【書類名】　明細書

【発明の名称】　競合調停装置および競合調停方法

【特許請求の範囲】

【請求項１】　複数のアプリケーションがアクセス対象に同時にアクセス要求をしたときに起こるアクセス競合を調停する装置において、

前記アクセス対象が必要とする一つ以上の競合発生要因を表すリソースごとに、現在アクセス可能なアプリケーションがどれかを示すリソース情報を保持するリソース情報記憶部と、

第１のアプリケーションから前記アクセス対象へのアクセス要求を受けた際に、前記リソース情報記憶部から前記リソース情報を取得し、現在アクセス権を持っている第２のアプリケーションが存在する場合は前記第１のアプリケーションと前記第２のアプリケーションのどちらにアクセス権を持たせるかを判定するリソースアクセス判定部と、

前記第１のアプリケーションが、前記アクセス対象が必要とする全ての前記リソースのアクセス権を取得したかを判定するアクセス対象アクセス判定部とを備えた競合調停装置。

【請求項２】　前記第１のアプリケーションからアクセス要求を受信するアプリケーションＩＦ部を新たに持つ請求項１に記載の競合調停装置。

【請求項３】　前記アクセス対象が論理デバイスの場合、前記アクセス対象アクセス判定部において、前記第１のアプリケーションが前記論理デバイスにアクセス可能な場合は前記論理デバイスを制御するデバイスドライバを呼び出す実行部を新たに持つ請求項１に記載の競合調停装置。

【請求項４】　前記実行部において、前記第１のアプリケーションが前記論理デバイスへのアクセス権を取得できなかった場合に、前記アプリケーションＩＦ部は前記第１のアプリケーションにエラーを通知する請求項３に記載の競合調停装置。

【請求項５】　前記論理デバイスが必要とするリソースとを対応付けしたデバイス情報を保持し、前記第１のアプリケーションからのアクセス要求を受信したときに必要なリソースを前記デバイス情報から取得する使用リソース取得部を新

たに備えた請求項１に記載の競合調停装置。

【請求項６】　前記リソースアクセス判定部において、アクセス競合が発生した場合にアプリケーションの優先度に基づいてアクセス権を取得させるアプリケーションを判定する請求項１に記載の競合調停装置。

【請求項７】　前記アプリケーションの優先度の情報を保持するアプリケーション情報記憶部を新たに持つ請求項６に記載の競合調停装置。

【請求項８】　前記アプリケーションの優先度が同じ場合に、先にアクセス要求をしたアプリケーションにアクセス権を取得させるか、後からアクセス要求をしたアプリケーションにアクセス権を取得させるかの情報をリソースごとにリソース情報に持たせた請求項６に記載の競合調停装置。

【請求項９】　アプリケーションから最初の要求であるアクセス開始要求を受信したときに競合調停をおこない、アクセス権を取得したかどうかの情報を前記アプリケーション情報に持たせ、前記アプリケーションが２回目以降にアクセス要求したときは、前記アプリケーション情報を参照してアクセス可能かどうかを判定する請求項７に記載の競合調停装置。

【請求項１０】　前記第２のアプリケーションは前記第１のアプリケーションにアクセス権を奪われた場合に、前記第２のアプリケーションから再度アクセス要求があったときに前記第２のアプリケーションにエラーを通知する請求項９に記載の競合調停装置。

【請求項１１】　前記第１のアプリケーションが最後にアクセス終了要求をしたときに、前記第１のアプリケーションにアクセス権を奪われていたために今までデバイスにアクセスできなかった第３のアプリケーションをアクセス可能とし、前記デバイスを前記第３のアプリケーションがアクセスするための状態に復元させる制御命令をデバイスドライバに通知する請求項９に記載の競合調停装置。

【請求項１２】　前記リソース情報記憶部は、多重アクセスを許すデバイスの場合はその該当機能のリソース情報を複数保持する請求項１に記載の競合調停装置。

【請求項１３】　前記リソース情報記憶部は、ある条件に基づいて多重アクセスを許すデバイスの場合は、その該当機能のリソース情報を保持する請求項１に

記載の競合調停装置。

【請求項１４】　一つのデバイスに対して複数のアプリケーションがアクセス要求をしたときに起こるアクセス競合を調停する方法において、

第１のアプリケーションからデバイスへのアクセス要求を受信する工程と、

アプリケーションのアクセス対象デバイスとそのデバイスが使用する機能を表すリソースとを対応付けしたデバイス情報を予め保持しており、前記第１のアプリケーションがアクセスする前記デバイスが使用するリソースを前記デバイス情報から取得する工程と、

リソースに対して現在アクセス権を持っているアプリケーションが存在するか、存在する場合はどのアプリケーションかを示す更新可能なリソース情報を保持し、必要に応じて更新する工程と、

前記リソース情報から前記第１のアプリケーションがアクセス対象としている前記デバイスが使用する前記リソースのリソース情報を取得し、現在アクセス権を持っている第２のアプリケーションとアクセス要求をしている前記第１のアプリケーションとで、どちらにアクセス権を持たせるかを判定する工程と、

アクセス対象の前記デバイスが使用する全ての前記リソースに対して、前記第１のアプリケーションがアクセス権を取得し、前記デバイスにアクセス可能かを判定する工程と、

デバイスにアクセス可能な場合は前記デバイスを制御するデバイスドライバを呼び出す工程とを備えた競合調停方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、複数のアプリケーションが起動するコンピュータシステム内において、複数のアプリケーションがアクセス対象に同時にアクセスする時に生じる競合の調停方法に関するものである。

【０００２】

【従来の技術】

複数のアプリケーションが起動するコンピュータシステム内では、複数のアプ

リケーションが一つのデバイスに同時にアクセス要求をすることにより競合が発生する。共有資源の排他制御方法としては、複数の計算機からその共有資源へのアクセス要求が来たときに、排他制御識別子を用いて競合調停をする方法が開示されている（例えば特許文献１参照）。

【０００３】

また一つのコンピュータシステム内において同時に複数のスレッドから記憶データにアクセス要求が来たときに、先勝ちにより競合調停をする方法も開示されている（例えば特許文献２参照）。

【０００４】

また一つのコンピュータシステム内において、調停機能を持たせて他のデバイスを制御する制御デバイスを設け、さらにアクセス要求に付与した優先度に基づいて競合調停をする方法も開示されている（例えば特許文献３参照）。

【０００５】

また一つの制御機器に対して複数の制御命令が来たときに、所定の条件に基づいて競合調停をする方法も開示されている（例えば特許文献４参照）。

【０００６】

【特許文献１】

特開２００２－１７５２８７号公報

【特許文献２】

特開平１０－１８７５２７号公報

【特許文献３】

特開２０００－２３１４５８号公報

【特許文献４】

特開２００１－３４６２７６号公報

【０００７】

【発明が解決しようとする課題】

しかし、このような従来の技術で、例えば一つのコンピュータシステム内で競合調停を行う場合では、競合調停を実際に存在するデバイス単位で行っていた。そのため実際にはデバイスは一つしかないが、ある条件の下では複数のアプリケ

ーションから同時にアクセスが可能である、またアプリケーションがある数までなら同時にアクセスが可能である、といった多重アクセスを許すデバイスにおいて競合調停を行っても、一つのアクセス要求しか受け付けず、デバイスの特性を十分に活かすことができなかった。また他のデバイスと配線結合され、単体ではＩ／Ｏポートを持たないデバイスが複数のデバイス間で共有されるような場合では、アプリケーション間で状態遷移表を用いて競合を回避するしかなく、アプリケーションやデバイスの構成に依存せざるを得ず、数が増えるに従い調停が困難であった。

【０００８】

本発明は、このようなデバイスの特性を活かしきるために提案された競合調停方法である。アクセス対象とするデバイスと、実際に競合調停を行うリソースとを分離して考えることにより、実際には一つしか存在しないが多重アクセスを許す機能を持つデバイスに対して、その機能を活かしきる競合調停を実現可能とすることを目的とする。

【０００９】

【課題を解決するための手段】

上記の目的を達成するために、一つのデバイスに対して複数のアプリケーションがアクセス要求をしたときに起こるアクセス競合を調停する装置において、第１のアプリケーションから第１のデバイスへのアクセス要求を受信するアクセス要求受信部と、アプリケーションのアクセス対象デバイスとそのデバイスが使用する機能を表すリソースとを対応付けしたデバイス情報を予め保持しており、前記第１のアプリケーションがアクセスする前記第１のデバイスが使用する第１のリソースを前記デバイス情報から取得する使用リソース取得部と、リソースに対して現在アクセス権を持っているアプリケーションが存在するか、存在する場合はどのアプリケーションかを示す更新可能なリソース情報を保持するリソース情報記憶部と、前記リソース情報記憶部から前記第１のアプリケーションがアクセス対象としている前記第１のデバイスが使用する前記第１のリソースのリソース情報を取得し、現在アクセス権を持っている第２のアプリケーションとアクセス要求をしている前記第１のアプリケーションとで、どちらにアクセス権を持たせ

るかを判定するリソースアクセス判定部と、アクセス対象の前記第１のデバイスが使用する全ての前記第１のリソースに対して、前記第１のアプリケーションがアクセス権を取得し、前記デバイスにアクセス可能かを判定するアクセス対象アクセス判定部と、デバイスにアクセス可能な場合は前記第１のデバイスを制御するデバイスドライバを呼び出す実行部とを備えた競合調停装置である。

【００１０】

【発明の実施の形態】

以下、本発明の実施の形態について、図面を参照しながら説明する。図１は、本発明の実施の形態における競合調停装置の構成の一例を示すブロック図である。図１において、競合調停装置は、アプリケーションＩＦ部１１と、アプリケーション情報記憶部１２と、使用リソース取得部１３と、リソース情報記録部１４と、リソースアクセス判定部１５と、アクセス対象アクセス判定部１６と、実行部１７とを備え、アプリケーションから受け取った要求に基づいて競合調停を処理し、対応するデバイスを使用するためにデバイスドライバの呼び出しを行う。

【００１１】

アプリケーションＩＦ部１１は、アプリケーションからアクセス開始要求を受信すると、使用リソース取得部１３にアクセス対象論理デバイスへのアクセス可否判定を要求し、アクセスを要求したアプリケーションの優先度をアプリケーション情報記憶部１２に格納処理する。また、アプリケーションからアクセス要求を受信すると、実行部１７を呼び出すことにより、アプリケーションがデバイスへのアクセスを行う。なお、アプリケーションがデバイスへのアクセス不可の場合は、実行部１７から受信したエラーをアプリケーションに通知する。更に、アプリケーションからアクセス終了要求を受信すると、使用リソース取得部１３にアプリケーションが保持していたアクセス権の解放を要求する。

【００１２】

アプリケーション情報記憶部１２は、アプリケーション毎にアプリケーション情報を格納している。アプリケーション情報は、アプリケーションＩＦ部１１が設定するアプリケーションの優先度と、アクセス対象の論理デバイス名と、実行部１７が設定する論理デバイスにアクセス可能か不可かの情報から構成されてい

る。

【００１３】

使用リソース取得部１３は、論理デバイスと前記論理デバイスが必要とするリソースとの関係を示したデバイス情報を保持しており、アプリケーションＩＦ部１１からアクセス開始要求が通知されると、前記デバイス情報を用いて、アクセス開始が要求されている論理デバイスが必要とするリソースを検出し、検出したリソースをリソースアクセス判定部１５に通知する。なお、１つの論理デバイスに対して複数のリソースが関係付けられてもよく、リソースアクセス判定部１５には、複数のリソースを同時に通知してもよいし、１つずつ順番に通知してもよい。また、論理デバイスとリソースとの関係については、図２を用いて後に説明する。

【００１４】

リソース情報記憶部１４は、全てのリソースに対して既にアクセス権を取得しているアプリケーションが存在するか否か、また、存在する場合は、アクセス権を取得しているアプリケーションを特定する情報をリソース情報として格納している。このリソース情報は、リソースアクセス判定部１５が、リソースへのアクセス権を保持するアプリケーションが変更になったと判断した場合に、更新設定を行う。

【００１５】

リソースアクセス判定部１５は、使用リソース取得部１３から通知されたリソースに対するリソース情報をリソース情報記憶部１４から取得し、既にアクセス権を取得しているアプリケーションが存在する場合は、そのアプリケーションとアクセス開始を要求しているアプリケーションとでどちらが優先かを、アプリケーション情報記憶部１２に格納してあるアプリケーションの優先度に基づいて判定する。判定結果として、アクセス権を取得するアプリケーションが変更になった場合は、リソース情報記憶部１４の対応するリソースのリソース情報を更新する。

【００１６】

アクセス対象アクセス判定部１６は、使用リソース取得部１３が検出したリソ

ースと、リソース情報記憶部１４から取得したリソース情報を用いて、アクセス開始を要求しているアプリケーションがアクセスしたい論理デイバスが使用する全てのリソースに対してアクセス権を取得しているか否かを判定し、全てのリソースに対してアクセス権を取得している場合に、論理デバイスへのアクセスが可能であると決定する。

【００１７】

実行部１７は、アクセス対象アクセス判定部１６から論理デバイスへのアクセスが可能か否かの情報を受け取り、受け取った情報に基づいて、アプリケーション情報記憶部１２に格納されているアプリケーション情報の更新を行う。また、アプリケーションＩＦ部１１からアクセス要求が通知されると、アプリケーション情報記憶部１２に格納されているアクセスを要求したアプリケーションがアクセス可能な場合は、対応するデバイスドライバを呼び出す。なお、アプリケーションがデバイスへのアクセス不可の場合は、実行部１７は、アプリケーションＩＦ部１１にエラーを通知する。

【００１８】

本発明ではリソースを物理デバイスが提供する機能と捉え、リソース単位で競合調停を行う。図２はアプリケーションのアクセス対象の論理デバイスとリソースと物理デバイスとの関係を示す構成図の一例である。図２ではＩ／Ｏポートを持った物理デバイスはＳＤＳＰ物理デバイスとＭＩＤＩ物理デバイスであり、スピーカー物理デバイスはＳＤＳＰ物理デバイスとＭＩＤＩ物理デバイスに配線されており、Ｉ／Ｏポートは持っていないため、スピーカー物理デバイスのみを制御するということはできない。さらにスピーカー物理デバイスはＳＤＳＰ物理デバイスとＭＩＤＩ物理デバイスとで共有されているため、アプリケーションＡＰ１がＳＤＳＰ物理デバイスにアクセスするときは、アプリケーションＡＰ２はＭＩＤＩ物理デバイスにはアクセスできず、また逆も起こりえる。このようにＳＤＳＰ物理デバイスとＭＩＤＩ物理デバイスに対して同時にアクセス要求がある場合は、アプリケーションからは見えないスピーカー物理デバイスに対してアクセス競合が発生する。そのためこのアクセス競合を回避するには、従来ではアプリケーション間で調停を行う必要があり、アプリケーションやデバイスの種類が増

えるに従い、調停が困難になる。そこで、本発明では先に述べたとおり、物理デバイスが提供する機能をリソースと捉え、そのリソース単位で競合調停を行う。

【００１９】

図２ではＳＤＳＰ物理デバイスの機能を表すＳＤＳＰリソースと、ＭＩＤＩ物理デバイスの機能を表すＭＩＤＩリソースと、スピーカー物理デバイスの機能を表すスピーカーリソースが存在する。またアプリケーションがアクセス対象とするのは論理デバイスである。論理デバイスは実際に使用する物理デバイスの機能、つまりリソースを必要とする。図２ではＳＤＳＰ論理デバイスはＳＤＳＰリソースとスピーカーリソースを必要とし、ＭＩＤＩ論理デバイスはＭＩＤＩリソースとスピーカーリソースを必要とする。そしてアプリケーションがアクセス対象とする論理デバイスにアクセス可能となるのは、その論理デバイスが必要とする全てのリソースに対してアクセス権が得られたときである。また本発明では個々のリソースに対してアクセス権が得られるかどうかは、先勝ちや後勝ちなどの決め方以外に、アプリケーションの優先度に応じて決めることができる。例えば図２において、アプリケーションＡＰ１がＳＤＳＰ論理デバイスにアクセス要求をし、アプリケーションＡＰ２がＭＩＤＩ論理デバイスにアクセス要求をした場合を考える。ＡＰ１とＡＰ２ではＡＰ２の方が優先度が高いとする。この場合、ＳＤＳＰリソースにはＡＰ１しかアクセス要求をしていないので、ＡＰ１がＳＤＳＰリソースのアクセス権を取得する。また同様にＭＩＤＩリソースにはＡＰ２しかアクセス要求をしていないので、ＡＰ２がＭＩＤＩリソースのアクセス権を取得する。スピーカーリソースについては、ＡＰ１とＡＰ２の両方のアクセス対象論理デバイスが必要としているため、競合が発生する。ここではＡＰ２の方が優先度が高いため、ＡＰ２がスピーカーリソースのアクセス権を取得できる。そしてＡＰ２はアクセス対象のＭＩＤＩ論理デバイスが必要とするリソース全てにおいてアクセス権が取得できたため、アクセスが可能となる。またＡＰ１はアクセス対象のＳＤＳＰ論理デバイスが必要とするリソースのうち、スピーカーリソースについてアクセス権を取得できなかったため、アクセス不可となる。

【００２０】

図３にアプリケーションのアクセス対象論理デバイスとリソースと物理デバイ

スとの構成の別の例を示す。図３は、物理デバイスとしては１回線しか存在しないが、実際には同時に３回線まで使用できるマルチコール対応の回線デバイスの場合を示している。この場合、アプリケーションのアクセス対象である論理デバイスおよび回線物理デバイスは１つしかないが、回線リソースは３つ存在する、と考える。そして回線論理デバイスは１つの回線リソースを必要とする。例えば図３においてアプリケーションＡＰ１、ＡＰ２、ＡＰ３、ＡＰ４がともに回線論理デバイスに対してアクセス要求をし、優先度はＡＰ１＞ＡＰ２＞ＡＰ３＞ＡＰ４の順でＡＰ１が最も高い場合を考える。まずＡＰ１が回線論理デバイスに対してアクセス要求をすると、ＡＰ１は３つある回線リソースのうち、１つ目の回線リソース３２のアクセス権を取得する。そして次にＡＰ２が回線論理デバイスにアクセス要求をすると、ＡＰ２は２つ目の回線リソース３３のアクセス権を取得する。次に最も優先度の低いＡＰ４が回線論理デバイスにアクセス要求をすると、ＡＰ３は３つ目の回線リソース３４のアクセス権を取得する。そこへＡＰ４より優先度の高いＡＰ３が回線論理デバイスにアクセス要求をすると、ＡＰ４とＡＰ３とで優先度を比較し、ＡＰ３の方が高いため、ＡＰ４は３つ目の回線リソース３４に対してアクセス不可となる。またＡＰ３は回線リソース３４へのアクセス権を取得する。そして、ＡＰ１、ＡＰ２、ＡＰ３が回線論理デバイスにアクセス可能となる。

【００２１】

また図４に別の構成例を示す。図４は、物理デバイスとしては１つしか存在しないが、実際には同じコーデックであれば録音機能と再生機能は同時に使用できるＳＤＳＰデバイスの場合を示している。この場合、アプリケーションのアクセス対象論理デバイスはＳＤＳＰ録音論理デバイスとＳＤＳＰ再生論理デバイスの２つが存在し、リソースはコーデック機能を表すＳＤＳＰコーデックリソースと、録音機能を表すＳＤＳＰ録音リソースと、再生機能を表すＳＤＳＰ再生リソースの３つが存在する、と考える。さらにＳＤＳＰコーデックリソースはコーデックが同じ場合は録音機能と再生機能を同時に使用できるため、２つ存在する、と考えるのではなく、コーデック情報を付加情報として持たせ、１つのみ存在する、と考える。そしてアクセス要求をしているアプリケーションが複数存在する場

合は、優先度が最も高いアプリケーションのコーデックを設定し、それと同じコーデックであればＳＤＳＰコーデックリソースにはアクセス可能とする。例えば図４においてアプリケーションＡＰ１、ＡＰ２がＳＤＳＰ録音論理デバイスにアクセス要求をし、ＡＰ３がＳＤＳＰ再生論理デバイスにアクセス要求をし、優先度はＡＰ１＞ＡＰ２＞ＡＰ３の順でＡＰ１が最も高く、またＡＰ１、ＡＰ２、ＡＰ３はともに同じコーデックＸでアクセス要求をする場合を考える。まずＡＰ１がＳＤＳＰ録音論理デバイスに対してアクセス要求をするとＡＰ１はＳＤＳＰコーデックリソースのアクセス権を取得し、Ｘに設定する。またＳＤＳＰ録音リソースに対してもアクセス権を取得する。次にＡＰ２がＳＤＳＰ録音論理デバイスにアクセス要求をすると、ＡＰ２はＳＤＳＰコーデックリソースに対しては、より優先度の高いＡＰ１と同じコーデックのためアクセス権を取得できるが、ＳＤＳＰ録音リソースについてはＡＰ１と競合するため、アクセス権を取得できない。次にＡＰ３がＳＤＳＰ再生論理デバイスにアクセス要求をすると、ＡＰ３はＳＤＳＰコーデックリソースに対しては、より優先度の高いＡＰ１と同じコーデックであるためアクセス権を取得でき、またＳＤＳＰ再生リソースに対しては競合が発生していないのでアクセス権を取得できる。つまりＡＰ１、ＡＰ３はともにアクセス対象論理デバイスが必要とする全てのリソースにアクセス可能なため、それぞれＳＤＳＰ録音論理デバイスおよびＳＤＳＰ再生論理デバイスにアクセス可能であるが、ＡＰ２はＳＤＳＰ録音リソースのアクセス権が取得できないため、ＳＤＳＰ録音論理デバイスにアクセス不可となる。

【００２２】

それでは次に本発明の詳細な処理手順を説明する。アプリケーションはまず始めにアクセス開始要求をし、デバイスにアクセス可能かどうかを判定して、可能な場合はデバイスに必要な設定等を行う。そして実際にデバイスにアクセスするときはアクセス要求をする。そして最後にアプリケーションが終了するなどで、デバイスへのアクセスも終了するときに、アクセス終了要求をし、デバイスの状態を初期化したり、場合によっては他のアプリケーションが使用するための設定をおこなったりする。

【００２３】

図５にアプリケーションのアクセス開始要求時の処理手順を示す。まずアプリケーションＩＦ部１１は、アプリケーションＡＰ１からアクセス対象論理デバイスＤＥＶ１へのアクセス開始要求とＡＰ１の優先度を受信する（ＳＴ１）。次にアプリケーションＩＦ部１１は、ＡＰ１のアプリケーション情報をアプリケーション情報記憶部１２に登録する（ＳＴ２）。アプリケーション情報には、アプリケーションの優先度、アクセス対象論理デバイス名、その論理デバイスにアクセス可能か不可かのフラグなどがあるが、このとき登録するアプリケーション情報はアプリケーションの優先度、アクセス対象論理デバイス名である。次にアプリケーションＩＦ部１１は、ＡＰ１からＤＥＶ１へのアクセス開始要求がきたことを使用リソース取得部１３に通知する。そして使用リソース取得部１３はＤＥＶ１が使用するリソースのＩＤを取得する（ＳＴ３）。これは使用リソース取得部１３が予め保持しているアクセス対象論理デバイスとそれが必要とするリソースＩＤとの関係を示したデバイス情報から取得する。そして使用リソース取得部１３は、取得したリソースＩＤをリソースアクセス判定部１５に通知する。そしてリソースアクセス判定部１５はそのリソースのリソース情報をリソース情報記憶部１４から取得する（ＳＴ３）。このリソース情報はリソースごとに存在し、現在アクセス権を取得しているアプリケーションを示している。アクセス権を取得しているアプリケーションが存在しない場合はＮＵＬＬを示している。また図３に示した例のように同一リソースが複数存在する場合は、同一リソースＩＤのリソース情報が複数存在する。次にリソースアクセス判定部１５はそのリソースに対してＡＰ１がアクセス権を取得できるかを判定する（ＳＴ４）。この処理の詳細なフローチャートを図６に示す。まず図３に示した例のように同一リソースが複数存在するかを、リソース情報記憶部１４を参照して、そのリソースＩＤのリソース情報が複数あるか調べることで判定する（ＳＴ１１）。存在する場合、つまり図３の例のような場合では、まず全てのリソースに対して、アクセス権を持っているアプリケーションが存在するかを、リソース情報記憶部１４に保持されているリソース情報を用いて調べる（ＳＴ１２）。そして全てのリソースに対して存在する場合は、アプリケーション情報記憶部１２に保持されているアプリケーション情報を参照して最も優先度の低いアプリケーションＡＰ２を検索する（

ＳＴ１３）。そしてＡＰ１とＡＰ２とでアプリケーション情報に設定されている優先度を比較し（ＳＴ１４）、もしＡＰ１の優先度が高い場合（ＳＴ１５）はそのリソース情報に対して、アクセス権を取得しているアプリケーションはＡＰ１であると設定する（ＳＴ１６）。またこのときＡＰ２のアクセス対象論理デバイスが必要とするリソースの少なくとも一つ以上に対してアクセス不可となるため、ＡＰ２はアクセス対象論理デバイスにアクセス不可となる。そこでアプリケーション情報記憶部１２に保持されているＡＰ２のアプリケーション情報をアクセス不可に設定する（ＳＴ１７）。またＳＴ１２において、空きのリソース（アクセス権を取得しているアプリケーションが存在しないリソース）が存在する場合は、そのリソースに、アクセス権を取得しているアプリケーションはＡＰ１であると情報を設定する。また、ＳＴ１５において、ＡＰ１よりＡＰ２の方が優先度が高い場合は、リソース情報の更新はおこなわない。またＳＴ１１において同一リソースが複数存在しない場合、つまり図２や図４の例のような場合は、そのリソースに対してアクセス権を取得しているアプリケーションＡＰ２が存在するか、リソース情報記憶部１４を参照して調べる（ＳＴ１８）。そして存在する場合は、アプリケーション情報記憶部１２からＡＰ２のアプリケーション情報を取得し（ＳＴ１９）、ＳＴ１４の処理を行う。以上ＳＴ１１からＳＴ１９までの処理をアクセス対象論理デバイスＤＥＶ１が必要とする全てのリソースに対して行う（ＳＴ２０）。このＳＴ２０の判定処理は使用リソース取得部１３が行う。ＳＴ１１からＳＴ１９までの処理はリソースアクセス判定部１５が行う。そして全てのリソースに対して処理が終了したら、使用リソース取得部１３は、そのＤＥＶ１が必要とする全てのリソースＩＤをアクセス対象アクセス判定部１６に通知し、アクセス対象アクセス判定部１６は、ＤＥＶ１が必要とする全てのリソースのアクセス権をＡＰ１が取得できたかを、リソース情報記憶部１４に保持されているリソース情報を参照して検索する（ＳＴ５）。そしてアクセス対象アクセス判定部１６はその結果を実行部１７に通知し、必要とする全てのリソースにアクセス可能な場合は、実行部１７はアプリケーション情報記憶部１２に保持されているＡＰ１のアプリケーション情報にアクセス可能であると設定する（ＳＴ６）。ＡＰ１からアクセス要求がきたときは、このアプリケーション情報に設定された

アクセス可能か不可かの情報を参照して判定する。次に実行部１７は、ＤＥＶ１のバスドライバを実行する（ＳＴ７）。またＳＴ５にてアクセス不可なリソースが存在する場合は、実行部１７は、ＡＰ１のアプリケーション情報にアクセス不可であると設定する（ＳＴ８）。

【００２４】

次に図７にアプリケーションのアクセス要求時の処理手順を示す。アプリケーションはアクセス要求をする前にアクセス開始要求をしており、そのときにアクセス対象論理デバイスへアクセス可能かどうかが判定され、アクセス可能／不可の情報がアプリケーション情報に保持されているため、その情報を用いて処理を行う。まずアプリケーションＩＦ部１１は、アプリケーションＡＰ１からアクセス対象論理デバイスＤＥＶ１へのアクセス要求を受信する（ＳＴ２１）。次にアプリケーションＩＦ部１１はアプリケーション情報記憶部１２に保持されているＡＰ１のアプリケーション情報を参照し、ＤＥＶ１にアクセス可能かどうかを調べる（ＳＴ２２）。そしてＤＥＶ１へのアクセスが可能と設定されていた場合は（ＳＴ２３）、アプリケーションＩＦ部１１は実行部１７を呼び出し、実行部１７は、ＤＥＶ１のデバイスドライバを実行する（ＳＴ２４）。またＳＴ２３にてＤＥＶ１にアクセス不可であると設定されていた場合は、アプリケーションＩＦ部１１は、ＡＰ１にエラーを返す（ＳＴ２５）。

【００２５】

次に図８にアプリケーションのアクセス終了要求時の処理手順を示す。まずアプリケーションＩＦ部１１は、アプリケーションＡＰ１からアクセス対象論理デバイスＤＥＶ１へのアクセス終了要求を受信する（ＳＴ３１）。次にアプリケーションＩＦ部１１は、ＡＰ１からＤＥＶ１へのアクセス終了要求がきたことを使用リソース取得部１３に通知し、使用リソース取得部１３はＤＥＶ１が必要とするリソースのＩＤを取得し、そのリソースＩＤをリソースアクセス判定部１５に通知する。そしてリソースアクセス判定部１５はリソース情報記憶部１４からそのリソース情報を取得する（ＳＴ３２）。次にＡＰ１がアクセス権を取得しているリソースに対しては、そのリソースを必要とするアクセス対象論理デバイスに対して、アクセス開始要求をしているがアクセス権を取得できていないアプリケ

ーションが存在しないか検索し、存在する場合はそのアプリケーションにリソースへのアクセス権を与える（ＳＴ３３）。これはＡＰ１がＤＥＶ１へのアクセスを終了することによって今までリソース競合が起き、アクセスが不可になっていたアプリケーションを、アプリケーションがそのことを意識せずに、つまりアプリケーションからみれば自動的にアクセス可能に設定するためである。これにより、例えばアプリケーションＡＰ２がアクセス対象論理デバイスＤＥＶ２にアクセス開始要求をし、実際にはまだＤＥＶ２にアクセスしていないときに、ＡＰ２より優先度の高いＡＰ１がＤＥＶ１にアクセス開始要求をしてきた場合を考える。ここで、ＤＥＶ１とＤＥＶ２はともにリソースＲを必要とする。始めにＡＰ２がＤＥＶ２にアクセス開始要求をしたときは、リソースＲはアクセス競合が起きていないため、ＡＰ２はＤＥＶ２へのアクセスが可能である。次にＡＰ１がＤＥＶ１に対してアクセス開始要求をすると、ＡＰ１の方が優先度が高いため、リソースＲのアクセス権はＡＰ１が取得し、ＡＰ２はＤＥＶ２にアクセス不可となる。しかしＡＰ２は実際にＤＥＶ２に対してアクセス要求をしていないため、アクセス不可になったことは知らない。そしてＡＰ１がＤＥＶ１にアクセスし、必要な処理を終えてアクセス終了要求をする。そしてリソースＲはＡＰ１から解放される。ＤＥＶ２はリソースＲを必要としている。そこでＡＰ２がリソースＲへのアクセス権を取得できればＡＰ２は自分がアクセス開始要求をして、実際にＤＥＶ２にアクセスするまでの間にＤＥＶ２へアクセス不可となっていたことを知る必要なく、ＤＥＶ２へアクセスでき、デバイスアクセスするときのアプリケーションの作りが簡潔になる。このＳＴ３３の処理の詳細なフローチャートを図９に示す。まずリソースアクセス判定部１５は、ＡＰ１のアクセス対象論理デバイスＤＥＶ１が必要とするリソース情報を参照して、ＡＰ１がアクセス権を取得しているか調べる（ＳＴ４１）。そして取得している場合は、リソースアクセス判定部１５は、そのリソースＲを使用リソース取得部１３に通知し、使用リソース取得部１３は、使用リソース取得部１３が保持しているデバイス情報を参照してそのリソースＲを必要とするアクセス対象論理デバイスを検索し、次にアプリケーション情報記憶部１２が保持しているアプリケーション情報を参照して、そのアクセス対象論理デバイスにアクセス開始要求をしているアプリケーションが存在

するか調べる（ＳＴ４２）。存在する場合は、使用リソース取得部１３は、そのアプリケーションＩＤをリソースアクセス判定部１５に通知する。そしてリソースアクセス判定部１５は、次にそのリソースＲを必要としているアプリケーションが複数存在するか判定する（ＳＴ４３）。そして複数存在する場合は、リソースアクセス判定部１５は、その中から最も優先度の高いアプリケーションＡＰ３を、アプリケーション情報記憶部１２が保持しているアプリケーション情報を参照して検索する（ＳＴ４４）。次にリソースアクセス判定部１５は、リソース情報記憶部１４が保持しているリソースＲのリソース情報に、アクセス権を取得しているアプリケーションはＡＰ３であると設定する（ＳＴ４５）。またＳＴ４３にてリソースＲを必要とするアプリケーションがアプリケーションＡＰ３のひとつしか存在しない場合は、ＳＴ４５にてＡＰ３がアクセス権を取得したと設定する（ＳＴ４５）。またＳＴ４３にてリソースＲを必要とするアプリケーションが存在しない場合は、アクセス権を取得しているアプリケーションがいないので、ＳＴ４５にてリソース情報にＮＵＬＬを設定する。またＳＴ４１にてそもそもＡＰ１がリソースＲのアクセス権を取得していない場合はＳＴ４２からＳＴ４５までの処理はおこなわない。これらＳＴ４１からＳＴ４５までの処理を、ＡＰ１のアクセス対象であるＤＥＶ１が必要とする全てのリソースに対して行う（ＳＴ４６）。このＳＴ４６の判定処理は使用リソース取得部１３が行う。次にＳＴ３３において新たにアクセス権を取得したリソースがあるアプリケーションＡＰ３が存在する場合、使用リソース取得部１３はＡＰ３が必要とする全てのリソースＩＤをアクセス対象アクセス判定部１６に通知し、アクセス対象アクセス判定部１６は、ＡＰ３のアクセス対象論理デバイスが必要とする全てのリソースのアクセス権を取得しているか調べる（ＳＴ３４）。そしてアクセス対象アクセス判定部１６はその結果を実行部１７に通知し、必要とする全てのリソースのアクセス権を取得しているアプリケーションＡＰ２が存在する場合は（ＳＴ３５）、実行部１７は、アプリケーション情報記憶部１２に保持されているＡＰ２のアプリケーション情報に、アクセス対象論理デバイスにアクセス可能である、と設定をする（ＳＴ３６）。さらに実行部１７は、デバイスドライバに対して、デバイスの設定をＡＰ２が使用する設定に変更するように要求し、デバイスドライバは設定を

変更する（ＳＴ３７）。ただしこれはＡＰ２がアクセス開始要求をして、デバイスの初期設定をしたときに、デバイスドライバがその設定を保持しておく必要がある。もしこの機能がないデバイスドライバの場合は、ＡＰ２がアクセス要求をしてきたときにデバイス設定が必要だというエラーコードをＡＰ２に返せばよい。またデバイスドライバがこの機能をサポートしている場合は自動復元が可能となる。ＳＴ３６、ＳＴ３７の処理を、ＳＴ３４にて必要とする全てのリソースのアクセス権を取得した、全てのアプリケーションに対して行う。最後に実行部１７はアプリケーションＩＦ部１１に対して処理が終了したことを通知し、アプリケーションＩＦ部１１は、アクセス終了要求をしてきたＡＰ１のアプリケーション情報をアプリケーション情報記憶部１２から削除する（ＳＴ３８）。またＳＴ３５にて必要とする全てのリソースのアクセス権を取得したアプリケーションが存在しない場合は、ＳＴ３６、ＳＴ３７の処理はおこなわずにＡＰ１のアプリケーション情報を削除する（ＳＴ３８）。

【００２６】

以上の処理の流れにより、図２の構成例の場合において具体的にどのようになるかを説明する。例えば図２において、既にＡＰ２がＳＤＳＰ論理デバイスにアクセス可能であるときに、ＡＰ１がＭＩＤＩ論理デバイスにアクセス開始要求をしたときを考える。ＡＰ１の方がＡＰ２より優先度が高いとする。まずＡＰ１が必要なＭＩＤＩリソースとスピーカーリソースのリソース情報を取得する。ＭＩＤＩリソース、スピーカーリソースともに同一リソースは１つずつしか存在しないため、次にこれらのリソースのアクセス権を既に取得しているアプリケーションを検索する。スピーカーリソースについてはＡＰ２が既にアクセス権を取得しているため、次にＡＰ１とＡＰ２の優先度を比較する。そしてＡＰ１の方が優先度が高いため、ＡＰ１はＭＩＤＩリソース、スピーカーリソースともにアクセス可能となり、ＭＩＤＩ論理デバイスに対してアクセス可能となる。またＡＰ２はスピーカーリソースにアクセス不可となったため、ＳＤＳＰ論理デバイスに対してアクセス不可となる。この情報をそれぞれのアプリケーション情報に設定する。次にＡＰ２がアクセス要求をすると、既にアプリケーション情報にアクセス不可と設定されているため、ＡＰ２にはエラーが返り、ＡＰ２はアクセス不可なこ

とを知る。次に、ＡＰ２がアクセス要求をする前に、ＡＰ１がアクセス終了要求をした場合を考える。まずＡＰ１がアクセス権を取得していたリソースとしてＭＩＤＩリソースとスピーカーリソースを取得する。次にＭＩＤＩリソースに対してはアクセス権を必要としているアプリケーションが存在しないため、ＭＩＤＩリソース情報にＮＵＬＬを設定する。またスピーカーリソースに対しては、ＡＰ２がアクセス権を必要としているため、スピーカーリソース情報に、ＡＰ２がアクセス権を取得したと設定する。もしＡＰ２以外にもスピーカーリソースを必要とするアプリケーションＡＰ３が存在した場合は、ＡＰ２とＡＰ３のアプリケーションの優先度を比較し、優先度の高いアプリケーションにアクセス権を取得させる。次にＡＰ２はスピーカーリソースのアクセス権を新たに取得することにより、必要とするリソース全てにアクセス可能となったため、ＳＤＳＰ論理デバイスにアクセス可能になり、ＡＰ２のアプリケーション情報にもアクセス可能であると設定される。そしてＳＤＳＰデバイスドライバがＡＰ２の設定に復元する機能がある場合は、次にＡＰ２がアクセス要求をしたときにＡＰ２はアクセス不可になっていたことを知る必要なく、ＳＤＳＰ論理デバイスにアクセスできる。つまりＳＤＳＰ物理デバイスとスピーカー物理デバイスにアクセスできる。

【００２７】

次に図３の構成例の場合において具体的な処理の流れを説明する。例えば図３においてアプリケーションＡＰ１、ＡＰ２、ＡＰ４が既に回線論理デバイスにアクセス可能であるときに、ＡＰ３が回線論理デバイスにアクセス開始要求をしてきたときを考える。優先度はＡＰ１＞ＡＰ２＞ＡＰ３＞ＡＰ４の順でＡＰ１が最も高いとする。まず回線リソースは３つ存在するため、その全てに対して、アクセス権を取得しているアプリケーションが設定されていないリソースがないか検索する。既にＡＰ１、ＡＰ２、ＡＰ４の３つのアプリケーションがそれぞれ回線リソースのアクセス権を取得しているため、空いている回線リソースは存在しない。そこで次にＡＰ１、ＡＰ２、ＡＰ４の中で最も優先度の低いアプリケーションを検索する。ＡＰ４が最も優先度が低い。次にＡＰ４とＡＰ３の優先度を比較し、ＡＰ３の方が優先度が高いため、ＡＰ４がアクセス権を取得していた回線リソースに対して、ＡＰ３がアクセス権を取得する。そしてＡＰ３は回線論理デバ

イスにアクセス可能となる。またＡＰ４はアクセス不可となる。次にＡＰ４より優先度の低いＡＰ５がアクセス開始要求をしてきた場合を考える。既に回線リソースは全て埋まっているため、ＡＰ１、ＡＰ２、ＡＰ３の中で優先度の最も低いアプリケーションを検索する。次に検索結果のＡＰ３とＡＰ５の優先度を比較し、ＡＰ５の方が優先度が低いため、ＡＰ５はアクセス不可となる。次にＡＰ４、ＡＰ５がともに回線論理デバイスにアクセス要求をする前に、ＡＰ２がアクセス終了要求をした場合を考える。回線リソースを必要としているアプリケーションとしてＡＰ４、ＡＰ５の２つが存在するため、ＡＰ４とＡＰ５の優先度を比較する。そして優先度の高いＡＰ４が、ＡＰ２がアクセス権を取得していた回線リソースのアクセス権を取得する。そしてＡＰ４は回線論理デバイスにアクセス可能となる。

【００２８】

次に図４の構成例の場合において具体的な処理の流れを説明する。例えば図４においてアプリケーションＡＰ１が既にＳＤＳＰ録音論理デバイスにアクセス可能であるときに、ＡＰ３がＳＤＳＰ再生論理デバイスにアクセス開始要求をしたときを考える。ＡＰ１の方がＡＰ３より優先度が高いとする。またＡＰ１とＡＰ３は同じコーデックＸを使用するとする。またＳＤＳＰコーデックリソースは付加情報を持ったリソースで、設定された状態と同じであれば多重アクセスを許す性質を持っている。まずＳＤＳＰコーデックリソースは既にアクセス権を取得しているＡＰ１が存在するため、ＡＰ１とＡＰ３の優先度を比較する。ＡＰ１の方が優先度が高いため、普通のリソースであればＡＰ３はアクセス権を取得できないが、ＳＤＳＰコーデックリソースは多重アクセスを許すリソースであるため、ＳＤＳＰコーデックリソースに設定されているコーデックの種別とＡＰ３の使用するコーデックの種別を比較する。そして同じコーデックＸであるため、ＡＰ３もＳＤＳＰコーデックリソースへのアクセス権を取得し、ＳＤＳＰ再生論理デバイスに対してアクセス可能となる。次にＡＰ２がＳＤＳＰ録音論理デバイスにアクセス要求をした場合を考える。ＡＰ２の優先度はＡＰ１とＡＰ３の間である。まずＳＤＳＰコーデックリソースに既にアクセス可能となっている最も優先度の高いアプリケーションＡＰ１と優先度を比較する。ＡＰ１の方が優先度が高いた

め、次にＳＤＳＰコーデックリソースに設定されているコーデックの種別とＡＰ２の使用するコーデックの種別を比較する。ここでＡＰ２はＡＰ１やＡＰ３とは異なるコーデックＹを使用する場合、ＡＰ２はＳＤＳＰコーデックリソースのアクセス権を取得できないことになり、ＳＤＳＰ録音論理デバイスにはアクセス不可となる。次にＡＰ１がアクセス終了要求をしたときを考える。まずＳＤＳＰ録音リソースに対しては、ＡＰ２が必要としているため、ＡＰ２にアクセス権を取得させる。次にＳＤＳＰコーデックリソースに対しては、ＡＰ２とＡＰ３が必要としているため、ＡＰ２とＡＰ３の優先度を比較する。そしてＡＰ２の方が優先度が高いため、ＡＰ２がＳＤＳＰコーデックリソースのアクセス権を取得する。またコーデックの種別を設定しなおす。次にＳＤＳＰコーデックリソースに新たに設定されたコーデックの種別とＡＰ３が使用するコーデックの種別を比較する。ここでＡＰ２とＡＰ３は異なるコーデックを使用するため、ＡＰ３はＳＤＳＰコーデックリソースにアクセス不可となり、ＳＤＳＰ再生論理デバイスに対してアクセス不可となる。

【００２９】

【発明の効果】

以上の説明のとおり本発明によれば、アプリケーションのアクセス対象のデバイスと実際に存在する物理デバイスと競合調停を行う対象のリソースとを分離して考えることで、デバイスの構成やデバイスの特性に合わせたリソース構成をとることができ、例えばアクセス数がある数までは多重アクセスを許す競合調停や、ある条件を満たせば多重アクセスを許す競合調停など、複雑な競合調停が実現可能となる。また例えば他のデバイスと配線結合され、単体ではＩ／Ｏポートを持たないデバイスが複数のデバイス間で共有されるような場合でも、そのデバイスへのアクセス競合調停が実現可能となる。

【００３０】

また例えば複数の物理デバイスを同時に制御することでアプリケーションの処理が実行される場合において、同時アクセスが必要な全ての物理デバイスにアクセス可能な場合は該当のデバイスドライバが呼び出され処理が実行されるが、一つでもアクセス不可なデバイスがある場合はアプリケーションにエラーが戻ると

いう仕組みが必要である。この場合に、アプリケーションのアクセス対象デバイスに対応するリソースを、同時にアクセスが必要な複数の物理デバイスとする構成をとることでこの仕組みを実現できる。

【００３１】

またアプリケーションに付与した優先度を用いて競合調停を行うことで、例えば多重アクセスを許すデバイスにおいては、優先度のより高いものからアクセスが可能となる、といった優先度に応じた複雑な排他制御をすることが可能となる。

【００３２】

またアプリケーションがデバイスにアクセス可能かどうかをアプリケーション情報に持たせ、デバイスへアクセス要求をしているアプリケーションの構成が変更するときにこのアプリケーション情報を更新することで、アプリケーションがデバイスへのアクセス要求をする度にリソースアクセス判定部およびアクセス対象アクセス判定部による競合調停をする必要がなくなり、処理の高速化が図れる。さらにアプリケーションがアクセス要求したときに、アクセスが可能であれば該当のデバイスドライバが呼び出され処理し、不可であればアプリケーションにエラーが戻るという仕組みにより、アプリケーションはアクセス要求をするときにエラーの場合の対処をするだけでよく、アプリケーション開発者の負担が軽減される。さらに、他のアプリケーションのアクセス要求により、デバイスへのアクセス権を奪われたアプリケーションが次にアクセスするまでに再びそのデバイスへのアクセスが可能になった場合、アプリケーションは一度アクセスが不可になったということを知ることなく、特別な処理をせずにデバイスへのアクセスをすることができる。

【図面の簡単な説明】

【図１】

本発明の実施の形態の構成例の概要を示す図

【図２】

アクセス対象論理デバイスとリソースと物理デバイスとの構成例を示す図

【図３】

アクセス対象論理デバイスとリソースと物理デバイスとの構成例を示す図

【図４】

アクセス対象論理デバイスとリソースと物理デバイスとの構成例を示す図

【図５】

アクセス開始要求時の処理手順を示すフローチャート

【図６】

アクセス開始要求時の詳細な処理手順を示すフローチャート

【図７】

アクセス要求時の処理手順を示すフローチャート

【図８】

アクセス終了要求時の処理手順を示すフローチャート

【図９】

アクセス終了要求時の詳細な処理手順を示すフローチャート

【符号の説明】

１１　アプリケーションＩＦ部

１２　アプリケーション情報記憶部

１３　使用リソース取得部

１４　リソース情報記憶部

１５　リソースアクセス判定部

１６　アクセス対象アクセス判定部

１７　実行部

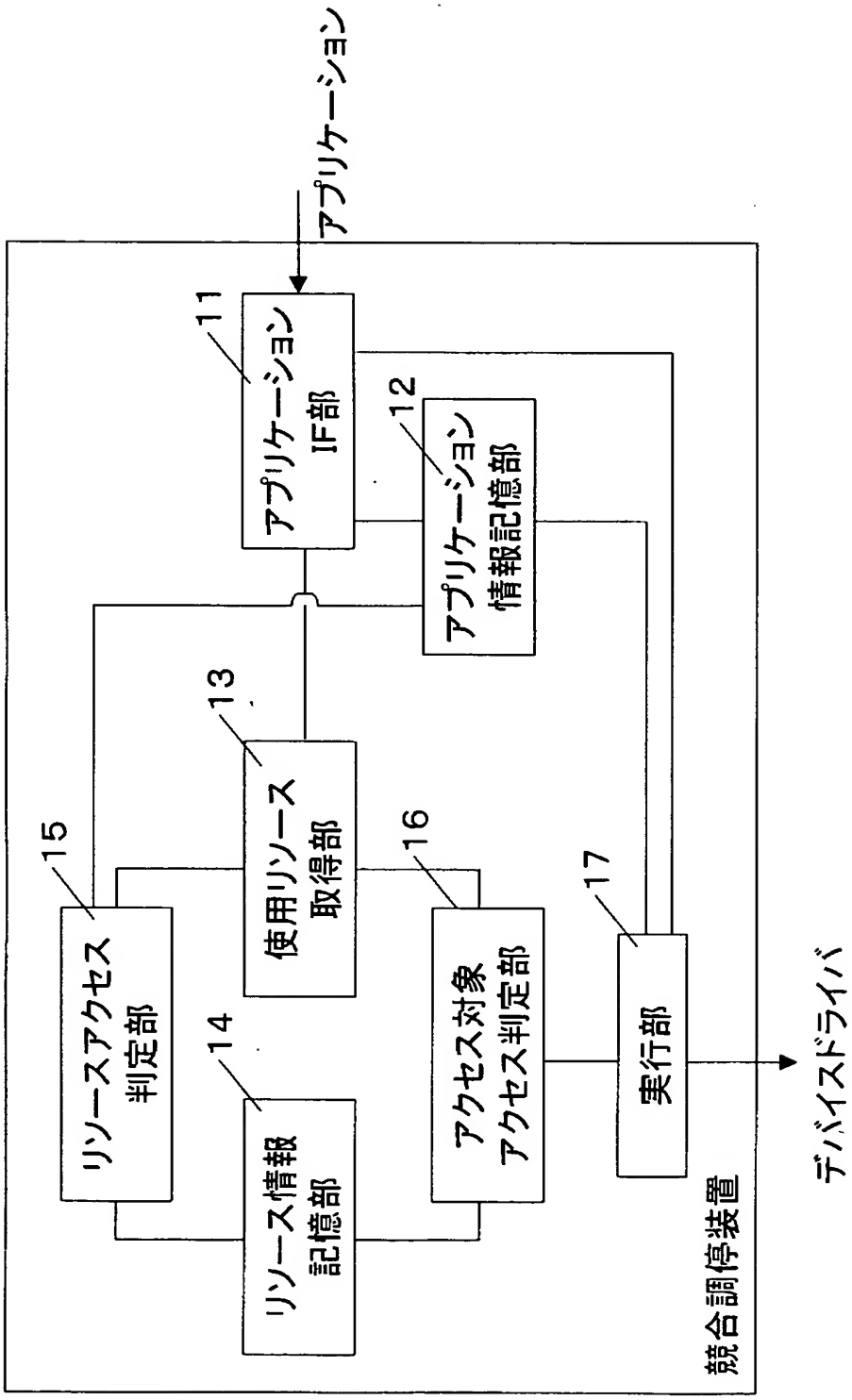
３２　１つ目の回線リソース

３３　２つ目の回線リソース

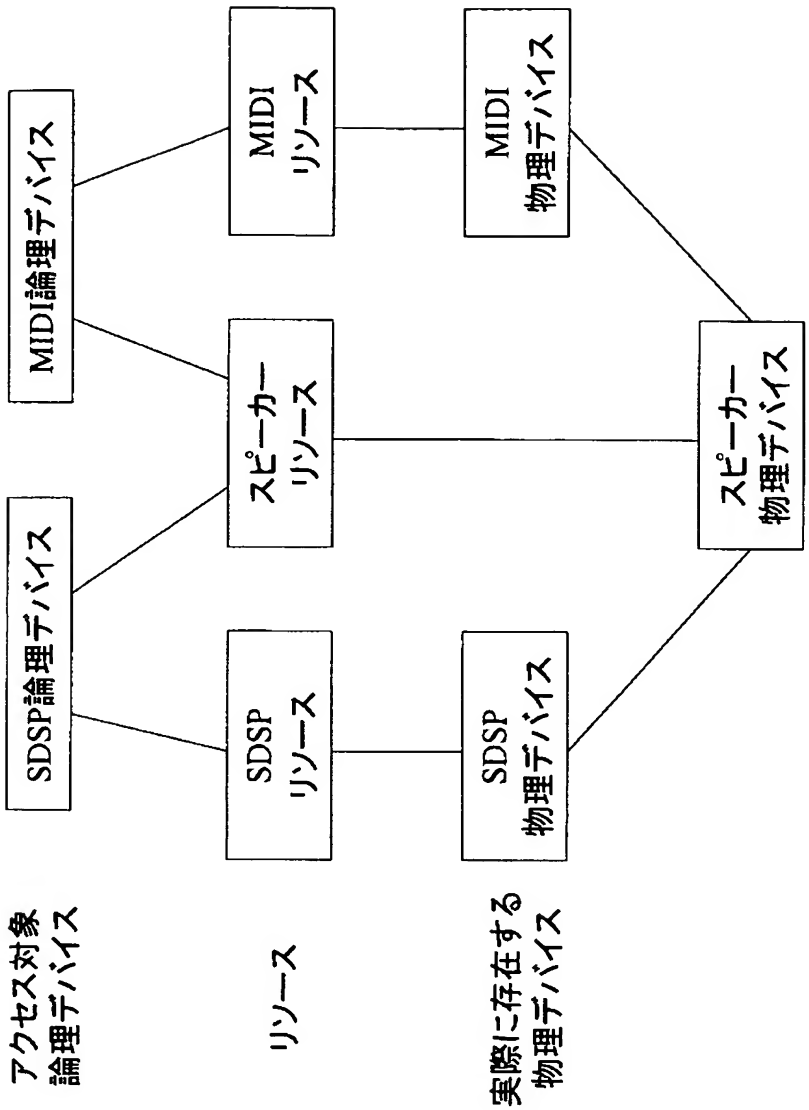
３４　３つ目の回線リソース

【書類名】　図面

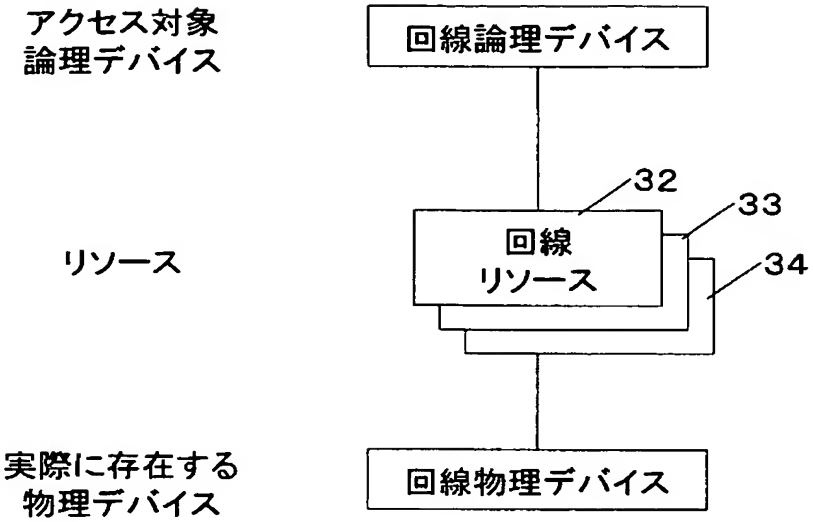
【図１】


アプリケーション
11
アプリケーション
IF部
12
アプリケーション
情報記憶部
13
使用リソース
取得部
15
リソースアクセス
判定部
14
リソース情報
記憶部
16
アクセス対象
アクセス判定部
17
実行部
デバイスドライバ
競合調停装置

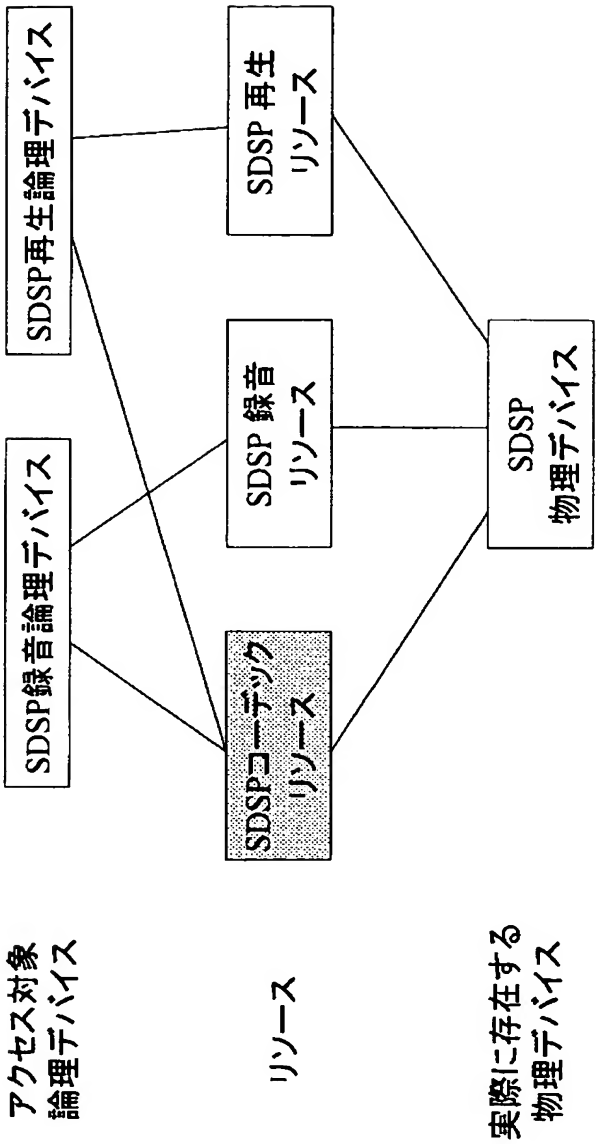
【図２】


MIDI論理デバイス
SDSP論理デバイス
MIDI
リソース
スピーカー
リソース
SDSP
リソース
MIDI
物理デバイス
スピーカー
物理デバイス
SDSP
物理デバイス
アクセス対象
論理デバイス
リソース
実際に存在する
物理デバイス

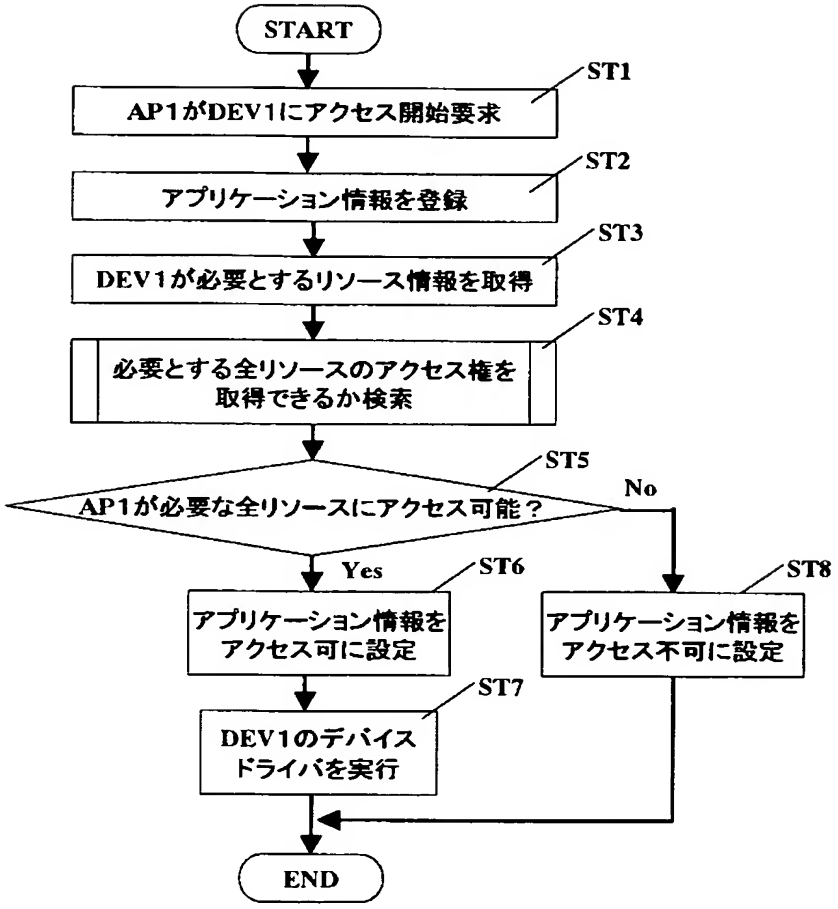
【図３】


アクセス対象
論理デバイス
回線論理デバイス
32
33
34
リソース
回線
リソース
実際に存在する
物理デバイス
回線物理デバイス

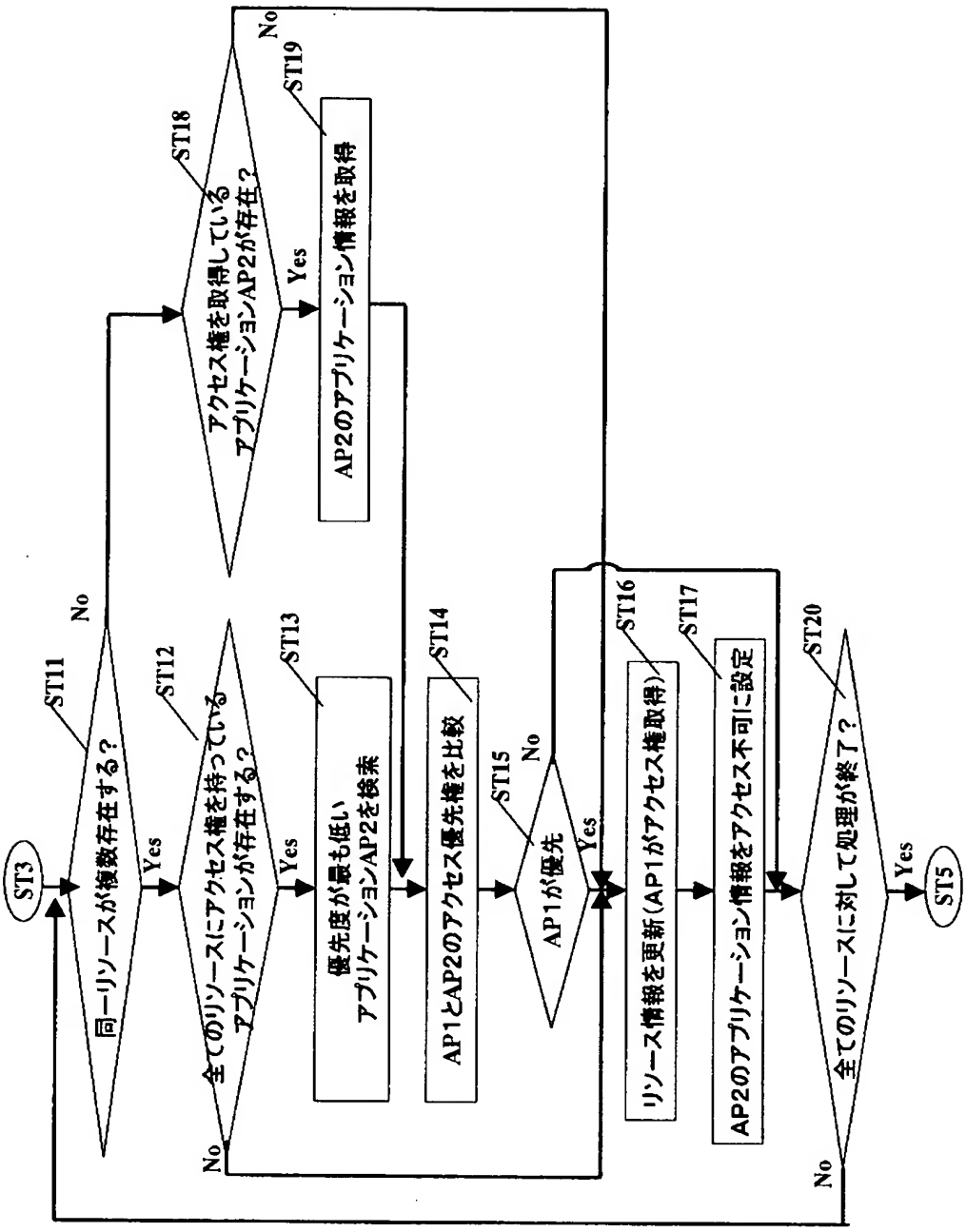
【図４】


SDSP再生論理デバイス
SDSP録音論理デバイス
SDSPコーデック リソース
SDSP 再生 リソース
SDSP 録音 リソース
SDSP 物理デバイス
アクセス対象 論理デバイス
リソース
実際に存在する 物理デバイス

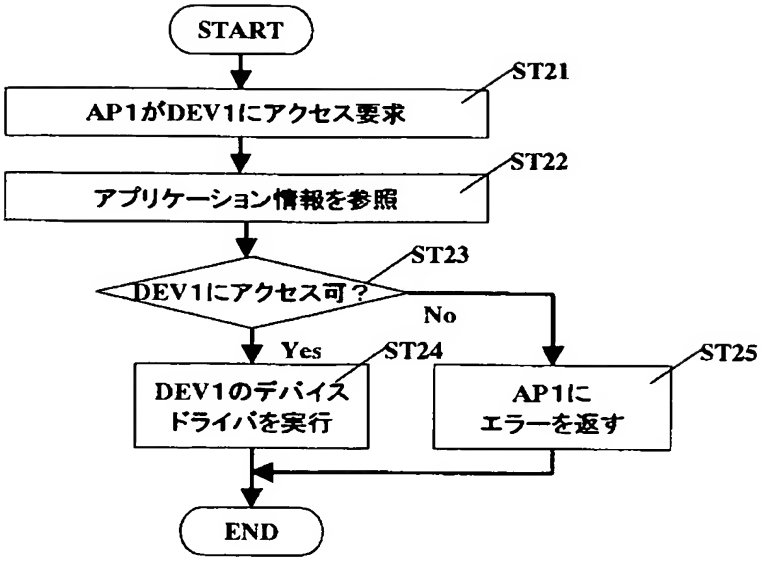
【図５】


START
ST1
AP1がDEV1にアクセス開始要求
ST2
アプリケーション情報を登録
ST3
DEV1が必要とするリソース情報を取得
ST4
必要とする全リソースのアクセス権を取得できるか検索
ST5
AP1が必要な全リソースにアクセス可能？
No
Yes
ST6
アプリケーション情報をアクセス可に設定
ST7
DEV1のデバイスドライバを実行
ST8
アプリケーション情報をアクセス不可に設定
END

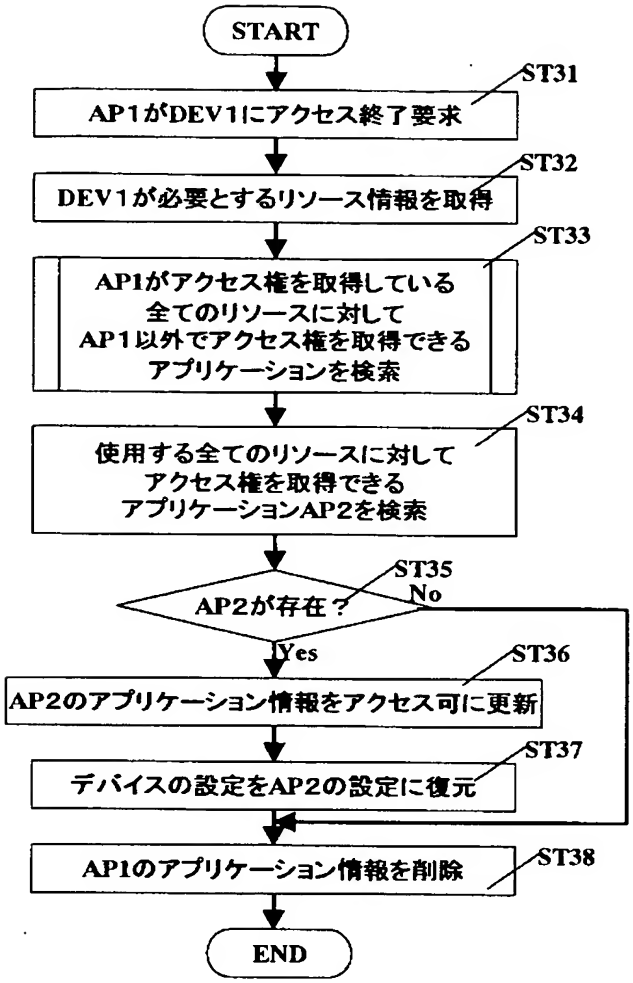
【図６】


ST3
ST11
同一リソースが複数存在する？
Yes
No
ST12
全てのリソースにアクセス権を持っている
アプリケーションが存在する？
Yes
No
ST13
優先度が最も低い
アプリケーションAP2を検索
ST14
AP1とAP2のアクセス優先権を比較
ST15
AP1が優先
Yes
No
ST16
リソース情報を更新（AP1がアクセス権取得）
ST17
AP2のアプリケーション情報をアクセス不可に設定
ST18
アクセス権を取得している
アプリケーションAP2が存在？
Yes
No
ST19
AP2のアプリケーション情報を取得
ST20
全てのリソースに対して処理が終了？
Yes
No
ST5

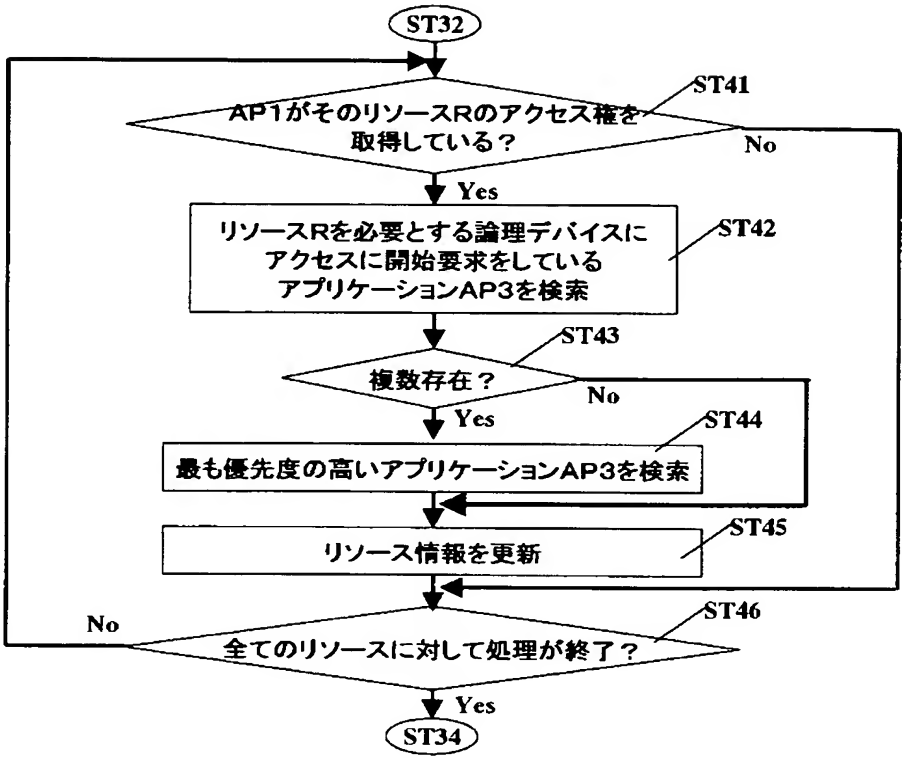
【図７】


START
ST21
AP1がDEV1にアクセス要求
ST22
アプリケーション情報を参照
ST23
DEV1にアクセス可？
No
Yes
ST24
DEV1のデバイス
ドライバを実行
ST25
AP1に
エラーを返す
END

【図８】


START
ST31
AP1がDEV1にアクセス終了要求
ST32
DEV1が必要とするリソース情報を取得
ST33
AP1がアクセス権を取得している
全てのリソースに対して
AP1以外でアクセス権を取得できる
アプリケーションを検索
ST34
使用する全てのリソースに対して
アクセス権を取得できる
アプリケーションAP2を検索
ST35
AP2が存在？
No
Yes
ST36
AP2のアプリケーション情報をアクセス可に更新
ST37
デバイスの設定をAP2の設定に復元
ST38
AP1のアプリケーション情報を削除
END

【図９】


ST32
ST41
AP1がそのリソースRのアクセス権を
取得している？
No
Yes
リソースRを必要とする論理デバイスに
アクセスに開始要求をしている
アプリケーションAP3を検索
ST42
ST43
複数存在？
No
Yes
ST44
最も優先度の高いアプリケーションAP3を検索
ST45
リソース情報を更新
ST46
No
全てのリソースに対して処理が終了？
Yes
ST34

【書類名】　要約書

【要約】

【課題】　アプリケーションのアクセス対象であるデバイスと競合調停を行う対象とを分離することで複雑な競合調停を可能にすることを目的としている。

【解決手段】　アプリケーションからアクセス要求を受信するアプリケーションＩＦ部と、アクセス対象のデバイスが使用するリソースを取得する使用リソース取得部と、そのリソースに対して現在アクセス権を持っているアプリケーションとアクセス要求をしているアプリケーションのどちらにアクセス権を持たせるか判定するリソースアクセス判定部と、アプリケーションがアクセス対象にアクセス可能かを判定するアクセス対象アクセス判定部と、デバイスを制御するデバイスドライバを呼び出す実行部とを備えた競合調停装置。

【選択図】　図１

特願２００２－３３１９０５

出　願　人　履　歴　情　報

識別番号　　［０００００５８２１］

１．変更年月日　　１９９０年　８月２８日
［変更理由］　　新規登録
住　所　　大阪府門真市大字門真１００６番地
氏　名　　松下電器産業株式会社